WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Thursday, April 01, 2004

Hide?	Set Name	Query	Hit Count
		DB=PGPB, USPT, EPAB, JPAB, DWPI, TDBD; PLUR=YES; OP=ADJ	
☐	L9	L8 not 16	5
☐	L8	15 and 13	12
☐	L7	15 and 12	77
☐	L6	14 and L5	7
☐	L5	717/$.ccls.	6159
☐	L4	L3 and (ActiveX)	15
☐	L3	L2 and ((interconnect$6 component$2) or (interconnect$6 part$2) or (interconnect$6 object$2))	42
☐	L2	L1 and container	961
☐	L1	(interconnect$6 or connect$6 or bind$3 or link$3) with (component$2 or object$2 or part$2) with software	9373

END OF SEARCH HISTORY

WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Thursday, April 01, 2004

Hide?	Set Name	Query	Hit Count
		DB=USPT, JPAB, EPAB, DWPI, TDBD; PLUR=YES; OP=ADJ	
☐	L7	(L5 and L6) and @pd > 20031209	0
☐	L6	((717/$)!.CCLS. or 705/$.ccls.) and @pd > 20031209	560
☐	L5	((time with estimat$6 with software with (develop$6 or creat$6 or writ$6 or generat$6))) and @pd > 20031209	3
		DB=USPT; PLUR=YES; OP=ADJ	
☐	L4	(L3 and (effort$2 same estimat$6 same software same development)) and @pd > 20031209	0
☐	L3	((((717/$)!.CCLS.))) and @pd > 20031209	260
		DB=USPT, JPAB, EPAB, DWPI, TDBD; PLUR=YES; OP=ADJ	
☐	L2	((effort$2 with estimat$6 with software)) and @pd > 20031209	1
		DB=USPT; PLUR=YES; OP=ADJ	
☐	L1	((effort$2 with estimat$6 with software with development$2)) and @pd > 20031209	1

END OF SEARCH HISTORY

L6: Entry 6 of 7 File: USPT Dec 17, 2002

DOCUMENT-IDENTIFIER: US 6496870 B1
TITLE: System, method and article of manufacture for collaboration with an application

Detailed Description Text (8):
Another technology that has function and capability similar to JAVA is provided by Microsoft and its ActiveX technology, to give developers and Web designers the wherewithal to build dynamic content for the Internet and personal computers. ActiveX runs only the so-called Wintel platform (a combination of a version of Windows and an Intel microprocessor), as contrasted with Java which is a compile once, run anywhere language.

Detailed Description Text (9):
ActiveX includes tools for developing animation, 3-D virtual reality, video and other multimedia content. The tools use Internet standards, work on multiple platforms, and are being supported by over one hundred companies. The group's building blocks are called ActiveX Controls, small, fast components that enable developers to embed parts of software in hypertext markup language (HTML) pages. ActiveX Controls work with a variety of programming languages including Microsoft's Visual C++, Borland's Delphi, Microsoft's Visual Basic programming system and, in the future, Microsoft's development tool for Java, code named "Jakarta." ActiveX Technologies also includes ActiveX Server Framework, allowing developers to create server applications. One of ordinary skill in the art will readily recognize that ActiveX and ActiveX components could be substituted for JAVA and its components as their use is described herein without undue experimentation to practice the invention.

Detailed Description Text (17):
Once obtained from its web page or local storage, the VJ Tool is itself initialized as shown in block 308 and made ready for interaction with a user, see block 310. Initialization of the applet includes initialization of the desktop (VJContainer) and the web page view (VJDocument). VJContainer and VJDesktop are tightly coupled. VJContainer is a container API. The details of applet initialization for VJ Tool are included in VJContainer and VJDesktop.

Detailed Description Text (20):
the source code the container editor for modifying properties of the container is presented below. When a new components is instantiated on either the physical or logical display, an optional customizer (edit) window is presented it a containerEditor method is defined for the component. The customizer (edit) window is defined in the template as a method corresponding to the customizer (edit) method. The method contains the logic facilitating the dynamic definition of properties of the component and the update of the properties based on user interaction with the customizer (edit) window.

Detailed Description Text (28):
FIG. 5 also shows the physical view 500 of the VJ desktop. The logical view 402 is in the foreground on the right hand side of the drawing, while the physical view 500 is in the background on the left side of the screen. Physical view 500 is

provided with palette containers 502 to 530 as follows: 502 is a select cursor button; 504 is a simple AWT label; 506 is a simple AWT text field; 508 is a simple AWT text area; 510 is a simple AWT button; 512 is a simple AWT checkbox; 514 is a simple AWT choice box; 516 is a simple AWT list; 518 is a simple AWT horizontal scroll bar; 520 is a simple AWT vertical scroll bar; 522 is a simple bar chart; 524 is a simple spreadsheet; 526 is a simple AWT panel; 528 is a calendar; and 530 is an animator.

Detailed Description Text (29):
Logical view 402 is provided with palette containers 540 to 572 as follows: 540 is a select cursor button; 542 is a VJ folder or container; 544 is an adder; 546 is a bicopy component. The bicopy component acts as a multiplexor so that whatever is input on one pin of a bicopy element is output on its other pins. 548 is a test for equals; 550 is a constant (e.g.--1 or 3.1457 or "abc"); 552 is a random number generator; 554 is a counter; 556 is an URL opener; 558 is a splitter used to connect input/output pins to components that are either input or output elements; 560 is a two-dimensional grapher; 562 is a three-dimensional grapher; 564 is a delta three-dimensional grapher; 566 is an URL text viewer; 570 forwards input from any one of an element's input pins to an outpin pin thereof; 572 is a calculator; and 574 is a sound player. AWT is the Abstract Windowing Toolkit and contains the objects utilized for providing basic windowing functions in accordance with a preferred embodiment.

Detailed Description Text (34):
Liveness also indicates that the environment is tested as new components are instantiated. This means that immediate connections between components are permitted on the fly. There is no requirement that all possible connections are tried first to insure that nothing blows up. The outcome of this capability is that design creation and trial are integrated so that the results can be observed virtually at the same time that the interconnection of components is made. As noted later herein in connection with the description of FIG. 16, this is analogous to building a hardware prototype or test board by interconnecting, disconnecting and moving electronic components around the test board while the power is left on. As will be shown by the following example of Fahrenheit and Centigrade conversion, there is immediate feedback of component status and value information. This makes VJ Tool, in essence, a WYSIWIG development environment.

Detailed Description Text (40):
FIG. 9 depicts an example of a VJScrollbar object 604b in the logical view 500. The physical view 402 illustrates a simple AWT horizontal scrollbar 902 having diamond shaped I/O pins or ports 904. The diamond shape indicates that the I/O pin is bi-directional or two way in operational nature; that is, they accept inputs or transmit outputs, as they are utilized. The I/O pins can be shaped like a triangle, rather than a diamond. If they are diamond shaped, then those I/O pins will handle an output or an input depending on the direction in which they point. I/O pins 904, which are sometimes referred to as "net pins", can be coded as follows: import java.awt.*; public class VJNetPin extends VJNode { // Attributes of this component static int instanceCount=0; static Image normalImage; static Image selectedImage; final static String out="out_np.gif"; final static String in="in_np.gif"; final static String port0_info="input or output and object"; final static String port0_name="Pin 0"; final static String url_name="netpin.html"; final static String info_name="Connects components inside a container to a pin of the container"; VJ vj; netpinEditor edit=null; VJContainer theContainer; boolean connected=false; boolean requested=false; int theLocation; int requestTime=0; int theConnection=-1; // Constructor public VJNetPin(VJv){ super(v); vj=v; } VJNode dup() { return null; } public static void getImages(GIFFactoryf){ normalImage=f.GetGIF(out); selectedImage=f.GetGIF(in); } public void setContainer(VJContainerc) { theContainer=c; } public void setConnection(int c) { theConnection=c; } // Component Initialization public void VJNetPinInit(intx_pt, int y_pt) { try { setNormalIcon(out); setSelectedIcon(in); setName("VJNetPin"); setComponent(null);

```
setComponentURL(url_name); setComponentInfo(info_name); VJNodeInit
(false,x_pt,y_pt,false); addPort
(port0_info,port0_name,VJPort.InputOutput,VJPort.SouthCenter); // Pin 0 setImages
(normalImage, selectedImage); //Pass references to the static images down to the
node nodeRect=new Rectangle(x_pt-3,y_pt-3, selectedImage.getWidth
(vj.theContainer.theDesktop.vp_w)+3,selectedImage. getHeight
(vj.theContainer.theDesktop.vp_w)+3); }catch(Exception e) { System.out.println
(e); } } public int componentID() {return 6; } public void disconnecting(int port)
{ connected=false; } public void connecting(int port) { connected=true; if
(requested) { vj.request(0,requestTime,this); requested=false; } } public void load
(String s) { } public String save() { return " "; } public void reset() {} public
void request(int port,int time) { // what if theConnection <0?
theContainer.requestOUT(theConnection,time); } public void requestIN(int time) { if
(connected)vj.request(0,time,this); else {requestTime=time; requested=true; } }
public void set(Object o,int port,int time) { if(theConnection>=0)
theContainer.setOUT(o,theConnection,time); //vj.request(0,request_index0++,this); }
public void setIN(Object o,int time) { vj.set(o,0,time,this); } public void
propertiesEditor() { if(edit==null){ edit=new netpinEditor((Frame)
(vj.theFrame),this); edit.pack(); edit.show(); } } public void init(){}; public
void start(){}; public void stop(){}; public void destroy(){}; } class netpinEditor
extends Frame { VJNetPin vjc; TextField tf; Button b; Button cancel; public
netpinEditor(Frame parent,VJNetPin c) { super("Pin Editor"); setLayout(new
BorderLayout()); add("North",new Label("Select a pin")); vjc=c; tf=new TextField
(new Integer(vjc.theLocation).toString()); add("Center",tf); b=new Button("OK");
cancel=new Button("Cancel"); Panel sp=new Panel(); sp.add(b); sp.add(cancel); add
("South",sp); } public boolean handleEvent(Event evt) { // System.out.println
(evt.toString()); switch(evt.id){ case Event.ACTION_EVENT: { if("OK".equals
(evt.arg)) { vjc.theLocation= (Integer.valueOf(tf.getText())).intValue
(); //vjc.theContainer.addNewPort(vjc,"fred","jim"); vjc.edit=null; dispose();
return true; } if("Cancel".equals(evt.arg)) { vjc.edit=null; dispose(); return
true; } return false; } default: return false; } } }
```

Detailed Description Text (73):
Decision block 1626 sends control back to polling if the pending connection has not yet been made. If the connection is completed, control for the ports involved is passed to decision block 1628 which decides if the attempted complete connection between the ports involved is valid. The type of ports involved and their functional affinity for each other will be checked and verified. Connections directly between an input port on one component will not be allowed to an input port on another component nor will a direct connection between an output port on a first container to an output port on a second container. Also forbidden are connections between ports on the same component regardless of their type.

Detailed Description Text (95):
FIG. 23 illustrates an edit screen for a folder component utilized to prepare a hierarchical component in accordance with a preferred embodiment. A Container 1 folder 2300 is instantiated in the logical desktop 2350 by dropping a folder component 2360 on the desktop. A customize or edit window 2310 pops up when the folder is instantiated. This edit window can be utilized to change the name of the folder and the number, location and type of active ports of the folder component. The Container 1 folder is logically represented by a window 2320 where the logic of the folder (hierachial component) can be created on the fly by adding or deleting components in the window 2320. The button 2330 is also shown in the physical desktop 2340.

Detailed Description Text (97):
FIG. 24 illustrates a hierarchical component creation customization in accordance with a preferred embodiment. The button 2400 which was originally in the logical desktop 2420 has been moved into the Container 1 folder 2410. The internal connectors 2430C and 2440C are represented in the Container 1 folder 2410 and refer

back to the logical desktop ports 2440L and 2430L of the folder 2450 in the logical desktop 2420. The ports can be used to transfer information into and out of a folder 2410. One of ordinary skill in the art will readily recognize that other components could be added to the Container 1 folder 2410 to create a more complex hierarchical component which could even include nested and recursive hierarchical components.

Detailed Description Paragraph Table (1):

```
import java.awt.*; import java.util.*; public class VJContainer extends VJNode
{ //Attributes of this component VJDesktop theDesktop = null; //if NOT null the
window (frame) associated with this container node //if null this is a primitive
node final static int cut = 1; final static int copy = 2; final static int paste =
3; int lastCommand = 0; final static String out = "out_nd.gif"; final static String
in = "in_nd.gif"; Vector port_info; Vector port_name; //final static String
port1_info = "output from container"; //final static String port1_name = "Pin 1";
final static String url_name = "container.html"; final static String info_name = "A
VJ Folder or container"; boolean open; int nodeCount; //the number of nodes static
int instanceCount = 0; //containerNode_container; //the panel in which contains a
hierarchcal node's nodes // //1) instantiate a new component on either physical or
logical display //and an optionai customizer (edit) window appears. //2) each
customizer (edit) window is defined in the template as a //method corresponding to
the customizer (edit) method. //3) Properties of the component are dynamically
updated based on //user interaction with the customizer (edit) window. // //
containerEditor edit; protected Vector nodes; //if null this is a primitive
node //otherwise the nodes contained in this hierarchical node int thisInstance;
static Image normalImage; static Image selectedImage; VJ vj; boolean outConnect[];
boolean request[]; boolean outRequest[]; int outRequestTime[]; int requestTime[];
VJNetPin thePin[]; int nextPort = 0; VJContainer theParent; //Constructor public
VJContainer(VJ v){ super(v); vj = v; } public static void getImages(GIFFactory f)
{ normalImage = f.GetGIF("out_nd.gif"); selectedImage = f.GetGIF("in_nd.gif"); }
VJNode dup() { return null; } //Component Initialization public void
VJContainerInit(int x_pt, int y_pt) { thisInstance = instanceCount++; setName(new
String("Container" +String.valueOf(thisInstance))); nodes = new Vector();
outConnect = new boolean[20]; request = new boolean[20]; outRequest = new boolean
[20]; outRequestTime = new int[20]; requestTime = new int[20]; thePin = new
VJNetPin[20]; for(int k=0; k<20; k++) { outConnect[k]=false; request[k]=false;
outRequestTime[k] = 0; requestTime[k] = 0; thePin[k]=null; } setNormalIcon
("out_nd.gif"); setSelectedIcon("in_nd.gif"); setComponentURL("container.html");
setComponentInfo("A simple VJ container"); VJNodeInit(true,x_pt,y_pt,false);
setImages(normalImage,selectedImage); //Pass references to the static images down
to the node theDesktop = new VJDesktop(vj,this); theDesktop.pack(); if
(instanceCount==1){ theDesktop.setTitle("VJ Desktop"); theDesktop.reshape
(10,30,400,640); theDesktop.show(); open = true; } else { theDesktop.setTitle
(getName()); theDesktop.reshape(instanceCount*20,instanceCount*20,400,460); }
port_info = new Vector(); port_name = new Vector(); nodeRect = new Rectangle(x_pt-
3,y_pt- 3,selectedImage.getWidth(vj.theContainer.theDesktop.vp_w)+3,
selectedImage.getHeight(vj.theContainer.theDesktop.vp_w)+3); } public void
addNewPort(VJNetPin addedPin, int i, String info, String name)
{ port_info.addElement(info); port_name.addElement(name); thePin[i] = addedPin;
addedPin.setConnection(i); addPort
(info,name,VJPort.InputOutput,addedPin.theLocation); // Pin 0 } public void
setParent(VJContainer p) { theParent = p; } public void request(int port,int time)
{ if(thePin[port]!=null) thePin[port].requestIN(time); else { System.out.println
("Pin connection problem"); } } public void requestOUT(int port,int time) { if
(outConnect[port]) vj.request(port,time,this); else { if(outRequest[port])
System.out.println("Losing previous out request"); outRequest[port] = true;
outRequestTime[port] = time; } } public int componentID() { return 500; } public
void disconnecting(int port) { if(port<20){ request[port] = false; outConnect[port]
= false; requestTime[port] = 0; } } public void connecting(int port) { if
(outRequest[port]) { outRequest[port] = false; vj.request(port,outRequestTime
```

```
[port],this); } } public void load(String s) { } public String save() { return
""; } public void set(Object o,int port,int time) { //System.out.println("set IN
port"+port); if(port<20){ thePin[port].setIN(o,time); } } public void setOUT(Object
o,int port,int time) { //System.out.println("set OUT port"+port); if(port<20)
{ vj.set(o,port,time,this); } } public void propertiesEditor() { if(!
theDesktop.isShowing().vertline..vertline.!theDesktop.isVisible()) theDesktop.show
(); if(edit==null && thisInstance > 0){ edit = new containerEditor((Frame)
(vj.theFrame),this); edit.pack(); edit.resize(6*32,6*32); edit.show(); } } public
void init(){ for(Enumeration e = nodes.elements() ; e.hasMoreElements() ;) { VJNode
vjn = (VJNode) e.nextElement(); vjn.init(); } }; public void start(){ if(open)
theDesktop.show(); for(Enumeration e = nodes.elements() ; e.hasMoreElements() ;)
{ VJNode vjn = (VJNode) e.nextElement(); vjn.start(); } }; public void stop(){ if
(open) theDesktop.hide(); for(Enumeration e = nodes.elements() ; e.hasMoreElements
() ;) { VJNode vjn = (VJNode) e.nextElement(); vjn.stop(); } }; public void destroy
(){}; public synchronized void addNode(Object o) { //System.out.println("Adding
node"); nodes.addElement(o); } public void doSelectAll(){ for(Enumeration e =
nodes.elements() ; e.hasMoreElements() ;) { VJNode vjn = (VJNode) e.nextElement();
vjn.setSelected(true); } theDesktop.vp_w.repaint(); vj.theDocument.repaint(); }
public void editComponent(){ for(Enumeration e = nodes.elements() ;
e.hasMoreElements() ;) { VJNode vjn = (VJNode) e.nextElement(); if(vjn.getSelected
()) vjn.propertiesEditor(); } } public void doCut(){ lastCommand = cut;
vj.nodePasteBoard.removeAllElements(); getSubnet(true); } public void doCopy()
{ lastCommand = copy; vj.nodePasteBoard.removeAllElements(); getSubnet(false); }
public void doPaste(){ int k,vCount=0; VJNode theSRCNode = null; if
(lastCommand==copy){ vCount = theDesktop.vp_w. container.nodes. size(); } for
(Enumeration e = vj.nodePasteBoard.elements() e.hasMoreElements() ;) { VJNode vjn =
(VJNode) e.nextElement(); VJNode vjn_c; if(lastCommand==copy) { vjn_c = vjn.dup();
if(vjn_c==null) { System.out.println("duplication failed in doPaste"); return; }
for(k=0; k<vjn.getNumberOfPorts() ;k++){ VJNode tn = vjn.getConnectingNode(k); if
(tn!=null && tn.getSelected()){ vjn_c.setConnectingPort(k,vjn.getConnectingPort
(k)); vjn_c.setToDraw(k,vjn.getToDraw(k)); } else { vjn_c.setConnectingPort(k,0);
vjn_c.setConnectingNode(k,null); vjn_c.setToDraw(k,false); } } if(vjn.isUINode)
vj.theDocument.clearLite(vj.theDocument.getGraphics(),vjn.comp. bounds()); } else {
```

Detailed Description Paragraph Table (4):

```
case VJPort.NorthLeft: x = xNode; y=yNode; break; case VJPort.NorthLeftCenter: x =
xNode+left; y=yNode; break; case VJPort.NorthCenter: x = xNode+center; y=yNode;
break; case VJPort.NorthRightCenter: x = xNode+right; y=yNode; break; case
VJPort.NorthRight: x = xNode+nRect.width; y=yNode; break; case VJPort.SouthLeft: x
= xNode; y=yNode+nRect.height; break; case VJPort.SouthLeftCenter: x = xNode+left;
y=yNode+nRect.height; break; case VJPort.SouthCenter: x =xNode+center;
y=yNode+nRect.height; break; case VJPort.SouthRightCenter: x =xNode+right;
y=yNode+nRect.height; break; case VJPOrt.SouthRight: x = xNode+nRect.width;
y=yNode+nRect.height; break; case VJPort.EastTop: x = xNode+nRect.width; y=yNode;
break; case VJPort.EastTopCenter: x = xNode+nRect.width; y=yNode+top; break; case
VJPort.EastCenter: x = xNode+nRect.width; y=yNode+middle; break; case
VJPort.EastBottomCenter: x = xNode+nRect.width; y=yNode+bottom; break; case
VJPort.EastBottom: x = xNode+nRect.width; y=yNode+nRect. height; break; case
VJPort.WestTop: x = xNode; y=yNode; break; case VJPort.WestTopCenter: x = xNode;
y=yNode+top; break; case VJPort.WestCenter: x = xNode; y=yNode+middle; break; case
VJPort.WestBottomCenter: x = xNode; y=yNode+bottom; break; default: x = xNode;
y=yNode+nRect.height; break;*/ } return new Dimension(x,y); } } VJDesktop is coded
as follows: import java.util.*; import java.awt.*; public class VJDesktop extends
Frame{ final String FILEMENU ="File"; final String NEWMENUITEM ="New"; final String
OPENMENUITEM ="Open..."; final String SAVEMENUITEM ="Save"; final String
SAVEASMENUITEM ="Save As..."; final String CLOSEMENUITEM ="Close"; final String
EXITMENUITEM ="Exit"; final String SEPARATORMENUITEM ="-"; final String
INFOMENUITEM ="View catalogue entry for current component..."; final String
EDITMENU ="Edit"; final String UNDOMENUITEM ="Undo"; final String EDITMENUITEM
```

```
="Edit Component"; final String CUTMENUITEM ="Cut"; final String COPYMENUITEM
="Copy"; final String PASTEMENUITEM ="Paste"; final String ALLMENUITEM ="Select
All"; final String CLEARMENUITEM ="Clear All"; final String APPLETMENU
="Environment"; final String RESETMENUITEM ="Reset all components"; final String
COMPILEMENUITEM ="Create standalone applet"; final String HELPMENU ="Help"; final
String ABOUTMENUITEM ="About Visual Java"; final String HELPTOPICSMENUITEM ="Help
Topics"; final static String EMPTY =""; Image in_controls[] = new Image[21]; Image
out_controls[] = new Image[21]; Image xy,wh; int current; int cnt; Toolbar tools;
int status=0; containerNode vp_w; VJContainer container; VJ applet_w; boolean
editorOpen=false; String user = EMPTY; String password = EMPTY; Vector
nodePasteBoard; public VJDesktop(VJ v, VJContainer n) { Panel centerPanel = new
Panel(); apple_w = v; container = n; MenuBar mb = new MenuBar(); Menu m = new Menu
(FILEMENU); m.add(new MenuItem(NEWMENUITEM)); m.add(new MenuItem(OPENMENUITEM));
m.add(new MenuItem(SAVEMENUITEM)); m.add(new MenuItem(SAVEASMENUITEM)); m.add(new
MenuItem(SEPARATORMENUITEM)); m.add(new MenuItem(CLOSEMENUITEM)); m.add(new
MenuItem(EXITMENUITEM)); mb.add(m); Menu m1 = new Menu(EDITMENU); m1.add(new
MenuItem(UNDOMENUITEM)); m1.add(new MenuItem(SEPAPATORMENUITEM)); m1.add(new
MenuItem(CUTMENUITEM)); m1.add(new MenuItem(COPYMENUITEM)); m1.add(new MenuItem
(PASTEMENUITEM)); m1.add(new MenuItem(ALLMENUITEM)); m1.add(new MenuItem
(CLEARMENUITEM)); m1.add(new MenuItem(EDITMENUITEM)); mb.add(m1); Menu m2 = new
Menu(APPLETMENU); m2.add(new MenuItem(INFOMENUITEM)); m2.add(new MenuItem
(RESETMENUITEM)); m2.add(new MenuItem(COMPILEMENUITEM)); mb.add(m2); Menu m3 = new
Menu(HELPMENU); m3.add(new MenuItem(ABOUTMENUITEM)); m3.add(new MenuItem
(SEPARATORMENUITEM)); m3.add(new MenuItem(HELPTOPICSMENUITEM)); mb.add(m3);
setMenuBar(mb); tools = new Toolbar(applet_w, true); doImages(); add("West",tools);
vp_w = new containerNode(applet_w, container); add("Center",vp_w); setBackground
(new Color(0,190,255)); } public void paint(Graphics g) { inth_off, v_off; if
(applet_w.isMicrosoft){ h_off = 0; v_off=0; } else { h_off=insets().left;
v_off=insets().top; } for(int i=0;i<cnt;i++) { int h = (i%12)*23+h_off; int v =
(i/12)*22+v_off; if(i == current && i <= (cnt-1)) g.drawImage(in_controls
[i],h,v,this); else g.drawImage(out_controls[i],h,v,this); } } public boolean
handleEvent(Event evt) { int h,v; if (evt.id == Event.ACTION_EVENT){ if (evt.target
instanceof MenuItem) { String label = (String)evt.arg; if (label.equals
(NEWMENUITEM)) { } else if (label.equals(OPENMENUITEM)) { } else if (label.equals
(SAVEMENUITEM)) { } else if (label.equals(SAVEASMENUITEM)) { } else if
(label.equals(CLOSEMENUITEM)) { if(container.thisInstance>0) { container.open =
false; hide(); } } else if (label.equals(EXITMENUITEM)) { } else if (label.equals
(INFOMENUITEM)) { } else if (label.equals(UNDOMENUITEM)) { doPaste(); } else if
(label.equals(CUTMENUITEM)) { doCut(); } else if (label.equals(COPYMENUITEM))
{ doCopy(); } else if (label.equals(PASTEMENUITEM)) { doPaste(); } else if
(label.equals(ALLMENUITEM)) { doSelectAll(); } else if (label.equals
(CLEARMENUITEM)) { doSelectAll(); doCut(); } else if (label.equals(EDITMENUITEM))
{ editComponent(); } else if (label.equals(RESETMENUITEM)) { } else if
(label.equals(COMPILEMENUITEM)) { } else if (label.equals(ABOUTMENUITEM)) { } else
if (label.equals(HELPTOPICSMENUITEM)) { } return true; } } if (evt.id ==
Event.WINDOW_DESTROY) { return true; } if (evt.id==Event.MOUSE_EXIT){ } if
(evt.id==Event.MOUSE_MOVE){ } if (evt.id==Event.MOUSE_DOWN){ } //System.out.print
(evt.toString()); return super.handleEvent(evt); } public void doSelectAll(){ if
(applet_w.loading) {System.out.println("Loading VJ"); return; }
container.doSelectAll(); } public void doCut(){ if(applet_w.loading)
{System.out.println("Loading VJ"); return; } container.doCut(); } public void
editComponent(){ if(applet_w.loading) {System.out.println("Loading VJ"); return; }
container.editComponent(); } public void doCopy(){ if(applet_w.loading)
{System.out.println("Loading VJ"); return; } container.doCopy(); } public void
doPaste(){ if(applet_w.loading) {System.out.println("Loading VJ"); return; }
container.doPaste(); } public void doImages() { GlFFactory factory = new GIFFactory
(applet_w); VJCInterface.getImages(factory); VJContainer.getImages(factory);
VJNetPin.getImages(factory); VJPlus.getImages(factory); VJBiCopy.getImages
(factory); VJEquais.getImages(factory); VJConstant.getImages(factory);
VJRandom.getImages(factory); VJCounter.getImages(factory); VJURLOpener.getImages
```

(factory); VJSplit.getImages(factory); tools.addItem(factory.GetGIF("in_cu.gif"), factory.GetGIF("out_cu.gif")); tools.addItem (VJContainer.selectedImage,VJContainer.normalImage); //tools.addItem (VJNetPin.selectedImage,VJNetPin.normalImage); tools.addItem (VJPlus.selectedImage,VJPlus.normalImage); tools.addItem (VJBiCopy.selectedImage,VJBiCopy.normalImage); tools.addItem (VJEquals.selectedImage,VJEquals.normalImage); tools.addItem (VJConstant.selectedImage,VJConstant.normalImage); tools.addItem (VJRandom.selectedImage,VJRandom.normalImage); tools.addItem (VJCounter.selectedImage,VJCounter.normalImage); tools.addItem (VJURLOpener.selectedImage, VJURLOpener.normalImage); tools.addItem (VJSplit.selectedImage,VJSplit.normalImage); } public String deskInfo(int ii) { switch(ii){ case 3: return "GGGG"; //VJButton.quick_info; default: return EMPTY; } } public String browserInfo(int ii){ switch(ii){ case 3: return "KKKKKKK"; //VJTextField.quick_info; default: return EMPTY; } } public void update (Graphics g) {

Detailed Description Paragraph Table (5):
paint(g); } public void drawNode(Image img,int x, int y){ vp_w.drawNode (img,x,y); } } class containerNode extends Panel { boolean marquee; boolean drag; boolean connecting; boolean disconnecting; int xbeg,xend,ybeg,yend; int left,top,right,bottom; int xleft,xright,ytop,ybottom; int current_i; int current_port; int portInfo; VJNode current_node; int current_comp_node; int downType; int inset_h, inset_v; boolean firstTime = true; boolean portInfoDrawn = false; boolean nodeInfoDrawn = false; static long lastTime=0; VJ app; VJContainer container; public containerNode(VJ v, VJContainer n) { super(); setLayout(null); disconnecting = false; app = v; container = n; } public boolean closeEnough(int fx, int fy,int x, int y,int epsilon){ return x <fx+epsilon && x >fx-epsilon && y <fy+epsilon && y >fy-epsilon; } public void doNodeSelection(VJNode vjn,boolean cntDwn,int x, int y){ if(!cntDwn) { beginDrag(x,y); if(!vjn.getSelected()) { app.theDocument.clearlight(app.theDocument.getGraphics()); container.resetSelected(); //reset all nodes in container to unselected vjn.setSelected(true); //select the current node app.theDocument.highlight (app.theDocument.getGraphics ()); } current_node = vjn; repaint(); }else{ if (vjn.getSelected()) { Rectangle r = vjn.nodeRect; vjn.setSelected(false); if (vjn.isUINode) app.theDocument.clearLite(app.theDocument.getGraphics(),vjn.comp. bounds()); getGraphics().clearRect(r.x-4,r.y-4,r.width+12,r.height+12); for (Enumeration e2 = container.nodes.elements(); e2.hasMoreElements() ;) { VJNode vjn2 = (VJNode) e2.nextElement(); if(vjn2.getSelected()) { repaint(); return; } } } else { vjn.setSelected(true); if(vjn.isUINode) app.theDocument.drawLite (app.theDocument.getGraphics(),vjn.comp. bounds()); repaint(); return; } } } public VJNode onNode(int x, int y){ for(Enumeration e = container.nodes.elements() e.hasMoreElements() ;) VJNode vjn = (VJNode) e.nextElement(); if (vjn.nodeRect.inside(x,y)){ //System.out. println("On node"+vjn.name); return vjn; } } return null; } public VJNode onPort(int x, int y){ Dimension d; current_port =-1; for(Enumeration e = container.nodes.elements(); e.hasMoreElements () ;){ VJNode vjn = (VJNode) e.nextElement(); for(int k=0; k<vjn.getNumberOfPorts () ;k++){ d = HOTSPOTS.getHotSpot(vjn.nodeRect,vjn.x,vjn.y,vjn.getPortLocation (k)) ; if(closeEnough(d.width,d.height,x,y,8)) { //System.out.println("On port ="+k+" of Node "+vjn.name); current_port = k; return vjn; } } } return null; } public boolean bigEnough(){ return xbeg-yend !=0 && ybeg-xbeg!=0; } public void beginConnection(int x, int y){ VJNode cn_beg; int cp_beg; connecting = true; if (disconnecting){ if(current_node == null .vertline..vertline. current_port == -1) { System.out.println("VJDesktop/doNodeSelection node or port not set"); return; //May need to do more work before/after return } cn_beg = current_node.getConnectingNode(current_port); cp_beg = current_node.getConnectingPort(current_port); if(cn_beg==null) System.out.println ("How can we be disconnecting ?"); xbeg = cn_beg.getXPt(cp_beg); ybeg = cn_beg.getYPt(cp_beg); xend = current_node.getXPt(current_port); yend = current_node.getYPt(current_port); } else { xbeg = x; ybeg = y; xend = x; yend = y;

```
Graphics g = getGraphics(); g.setXORMode(Color.white); g.drawLine(xbeg, ybeg, xend,
yend); } } public void doConnection(int x, int y){ //System.out.println("DO
CONNECTION"); Graphics g = getGraphics(); g.setXORMode(Color.white); g.drawLine
(xbeg, ybeg, xend, yend); xend = x; yend = y; g.drawLine(xbeg, ybeg, xend ,yend); }
public void endConnection(int x, int y){ int srcPort; VJNode
srcNode; //System.out.println("END CONNECTION"); Graphics g = getGraphics();
g.setXORMode(Color.white); g.drawLine(xbeg, ybeg, xend, yend); xend = x; yend = y;
connecting = false; srcNode = current_node if(srcNode==null) System.out.println
("srcNode =null?"); srcPort = current_port; current_node=onPort(x,y); VJNode yj_c =
srcNode.getConnectingNode(srcPort); if(vj_c==null) System.out.println("vj_c
=null?"); int vj_p = srcNode.getConnectingPort(srcPort); if(disconnecting &&
current_node==null){ int xb = srcNode.getXPt(srcPort); int xe = vj_c.getXPt
(vj_c.getConnectingPort(srcPort)); int yb = srcNode.getYPt(srcPort); int ye =
vj_c.getYPt(vj_c.getConnectingPort(srcPort)); if(xb < xe) { if(yb < ye) g.clearRect
(xb-2, yb-2, xe - xb+4, ye - yb+4); else g.clearRect(xb-2, ye-2, xe - xb+4, yb -
ye+4); } else { if(yb < ye) g.clearRect(xe-2, yb-2, xb - xe+4, ye - yb+4); else
g.clearRect(xe-2, ye-2, xb - xe+4, yb - ye+4); } vj_c.disconnecting
(srcNode.getConnectingPort(srcPort)); srcNode.disconnecting(srcPort);
vj_c.setConnectingNode(srcNode.getConnectingPort(srcPort),null);
vj_c.setConnectingPort(vj_p,0); vj_c.setXPt(vj_p,0), vj_c.setYPt(vj_p,0);
vj_c.setToDraw(vj_p,false); srcNode.setToDraw(srcPort,false);
srcNode.setConnectingNode(srcPort,null); srcNode.setConnectingPort(srcPort,0);
srcNode.setXPt(srcPort,0); srcNode.setYPt(srcPort,0); disconnecting = false;
return; } if(current_node!=null && compatible
(srcNode,srcPort,current_node,current_port)) { if(disconnecting){ int xb =
srcNode.getXPt(srcPort); int xe = vj_c.getXPt(vj_p); int yb = srcNode.getYPt
(srcPort); int ye = vj_c.getYPt(vj_p); if(xb < xe) { if(yb < ye) g.clearRect(xb-2,
yb-2, xe - xb+4, ye - yb+4); else g.clearRect(xb-2, ye-2, xe - xb+4, yb - ye+4); }
else { if(yb < ye) g.clearRect(xe-2, yb-2, xb - xe+4, ye - yb+4); else g.clearRect
(xe-2, ye-2, xb - xe+4, yb - ye+4); } vj_c.setConnectingNode(vj_p,current node);
vj_c.setConnectingPort(vj_p, current_port); vj_c.setToDraw(vj_p,true);
current_node.setConnectingNode(current_port,vj_c); current_node.setConnectingPort
(current_port,vj_p); current_node.setXPt(current_port,xend); current_node.setYPt
(current_port,yend); current_node.resetToDraw(current_port); vj_c.connecting(vj_p);
current_node.connecting(current_port); srcNode.disconnecting(srcPort);
srcNode.resetToDraw(srcPort); srcNode.setConnectingNode(srcPort,null);
srcNode.setConnectingPort(srcPort,0); srcNode.setXPt(srcPort,0); srcNode.setYPt
(srcPort,0); disconnecting = false; g.drawLine(xbeg, ybeg, xend, yend); return; }
srcNode.setConnectingNode(srcPort,current_node); srcNode.setConnectingPort
(srcPort,current_port); srcNode.setXPt(srcPort,xbeg); srcNode.setYPt(srcPort,ybeg);
srcNode.setToDraw(srcPort,true); current_node.setConnectingNode
(current_port,srcNode); current_node.setConnectingPort(current_port,srcPort);
current_node.setXPt(current_port,xend); current_node.setYPt(current_port,yend);
current_node.connecting(current_port); srcNode.connecting(srcPort); g.drawLine
(xbeg, ybeg, xend, yend); } } public void doAllConnects(){ }
```

Detailed Description Paragraph Table (7):

```
current_node = onNode(e.x,e.y); if(current_node!=null&&!nodeInfoDrawn)
{ EraseNodeInfo(); DrawNodeInfo(); nodeInfoDrawn = true; } else if
(current_node==null&&nodeInfoDrawn) { EraseNodeInfo(); nodeInfoDrawn = false; }
current_node = onPort(e.x,e.y); if(current_port>=0&&!portInfoDrawn) { ErasePortInfo
(); DrawPortInfo(); portInfoDrawn = true; portInfo = current_port; } else if
(portInfoDrawn&¤t_port!=portInfo) { ErasePortInfo(); portInfoDrawn = false; }
return true; case Event.KEY_PRESS: if(e.key==127) container.theDesktop.doCut(); if
(e.key==4) container.DUMP(); if(e.controlDown()){ switch (e.key){ case 3:
container. theDesktop.doCopy(); break; case 5: container.theDesktop.editComponent
(); break; case 24: container.theDesktop.doCut(); break; case 22:
container.theDesktop.doPaste(), break; case 1: container.theDesktop.doSelectAll();
break; default. break; } } return true; case Event.MOUSE_DOWN: if((e.when -
lastTime)<1000) { lastTime = e.when; System.out.println("DC");
```

container.editComponent(); return false; } lastTime = e.when; current_node = onNode (e.x,e.y); if(current_node!=null){ doNodeSelection(current_node e.controlDown (),e.x,e.y); } else { current_node = onPort(e.x,e.y); if(current_node!=null){ if (isConnected (current_node,current_port)) { //System.out.println("BEGIN DISCONNECTING"); disconnecting=true; } connecting=true; } else { if(!(e.controlDown ())){ app.theDocument.clearlight(app.theDocument.getGraphics()); container.resetSelected(); } current_comp_node = container.theDesktop.tools.getCurrent(); //System.out.println ("current_selection"+current_comp_node); if(current_comp_node>0) { //Add New component VJNode theNode=null; switch(current_comp_node){ case 1: VJContainer vjcnt = new VJContainer(app); vjcnt.setParent(container); vjcnt.VJContainerInit(e.x,e.y); theNode = vjcnt; break; // case 2: VJNetPin vjnp = new VJNetPin(app); // vjnp.setContainer(container); // vjnp.VJNetPinInit(e.x,e.y); // theNode = vjnp; // break; case 2: VJPlus vjp = new VJPlus(app); vjp.VJPlusInit(e.x,e.y); theNode = vjp; break; case 3: VJBiCopy vjb = new VJBiCopy(app); vjb.VJBiCopyInit(e.x,e.y); theNode = vjb; break; case 4: VJEquals vje = new VJEquals(app); vje.VJEqualsInit (e.x,e.y); theNode = vje; break; case 5: VJConstant vjc = new VJConstant(app); vjc.VJConstantInit(e.x,e.y); theNode = vjc; break; case 6: VJRandom vjr = new VJRandom(app); vjr.VJRandomInit(e.x,e.y); theNode = vjr; break; case 7: VJCounter vjct = new VJCounter(app); vjct.VJCounterInit(e.x,e.y); theNode = vjct; break; case 8: VJURLOpener vju = new VJURLOpener(app); vju.VJURLOpenerInit(e.x,e.y); theNode = vju; break; case 9: VJSplit vjs = new VJSplit(app); vjs.VJSplitInit(e.x,e.y); theNode = vjs; break; case 10: theNode = null; break; } if(theNode!=null) { theNode.init(); theNode.propertiesEditor(); if(!e.controlDown()) { container.resetSelected(); } container.addNode((Object)theNode); theNode. setSelected(true); container.theDesktop.tools.setCurrent(0); current_comp_node = container.theDesktop.tools.getCurrent(); container.theDesktop.tools.repaint (); } } } if(connecting) { beginConnection(e.x,e.y); } else beginMarquee (e.x,e.y); } return true; case Event.MOUSE_UP: //System.out.println("UP"); if (connecting) { endConnection(e.x,e.y); repaint(); return true;} if(marquee) { endMarquee(e.x,e.y); repaint(); return true;} if(drag) { endDrag(e.x,e.y); repaint(); return true;} return true; case Event.MOUSE_DRAG: //System.out.println ("DRAG"); if(connecting) { doConnection(e.x,e.y); return true; } if(marquee) { doMarquee(e.x,e.y); return true; } if(drag) { doDrag(e.x,e.y); return true; } return true; default: //System.out.println("Other event"+e.toString()); return false; } } void DrawPortInfo(){ Graphics g = getGraphics(); VJNode vjn; if (current_node==null .vertline..vertline. current_port<0) return; g.drawString (current_node.name+"Pin:"+current_port+" "+current_node.getPortInfo (current_port),30,40); } void ErasePortInfo(){ Graphics g = getGraphics(); g.clearRect(30,29,bounds().width,15); } void DrawNodeInfo(){ Graphics g = getGraphics(); if(current_node == null) return; g.drawString (current_node.name+":"+current_node.getNodeInfo(),30,2 2); } void EraseNodeInfo() { Graphics g = getGraphics(); g.clearRect(30,12,bounds().width,13); } void Error (String error) { System.out.println("Node:"+error); System.out.flush(); } } VJNode is coded as follows: import java.awt.*; import java.awt.image.*; import java.util.*; //A class that is used to represent both primitive and hierarchical VJ nodes abstract class VJNode extends VJCore { //Class attributes private final static String getPortNameError = "get port name error"; private final static String getPortInfoError = "get port info error"; private final static String noQuickInfo = "no quick info available"; private final static String noAuthorInfo = "no author info available"; private final static String noExpirationDate = "no expiration date"; private final static String noVersionInfo = "no version info"; private final static String noCostInfo = "no cost info"; private final static String noName = "no name"; private final static String noComponentURL = "no URL"; private final static String noPortName = "no port name"; //Attributes private String info_url; private String quick_info; private String author_info; private String version_info; private String cost_info; private String expiration_date; private String componentURL; private Vector port_name; private Vector port_info; private Vector port_type; private Vector port_location; private Vector XPts; private Vector YPts; private int numberOfPorts=0; private Image normalImage=null; private Image selectedImage=null;

private String normal = null; private String selected = null; private GIFFactory factory=null; boolean isContainer; //true if the node is hierarchical boolean isSelected; //true if the node is currently selected boolean isUINode; //true if the node has a user interface (exits on the web page) int x; //the x position in the parent container int y; //the y position in the parent container Vector drawFromPort; int portCount; //the number of ports the node has VJ vj; //a reference to the VJ applet Rectangle nodeRect; //the rectangle accociated with this nodes image String name; public VJNode(VJ v){ super(v); vj = v; port_name = new Vector(); port_type = new Vector();

Detailed Description Paragraph Table (9):
import java.util.*; import java.awt.*; class VJDocument extends container { VJ app; VJContainer container; VJNode current_i; int current_ui_comp; static int xOffset = 40; // used to define the position on the desktop static int yOffset = 100; // at which the new icon corresponding to a // component is to be placed. int top,left,right,bottom; int ytop,xleft,xright,ybottom; boolean grow; int growType; public VJDocument(VJ v,VJContainer c){ // Set the environment variables super(v); app = v; container = c; current_ui_comp = 0; current_i = null; // Initial Layout for the Applet setLayout(null); } void marqueeAction(boolean cntrDwn,Rectangle r,int x,int y){ current_ui_comp = app.uiTools.getCurrent(); if(current_ui_comp>0 && bigEnough()) { Graphics g = getGraphics(); VJNode vjn = newComponent(cntrDwn,x,y); if(vjn!=null){; if(!cntrDwn) clearlight(g); current_ui_comp = 0; vjn.setSelected (true); container.theDesktop.vp_w.repaint(); drawLite(g,r); app.uiTools.setCurrent (0); app.uiTools.repaint(); } return; } boolean emptySelect; emptySelect = true; for(Enumeration e = app.theContainer.nodes.elements(); e.hasMoreElements();) { VJNode vjn = (VJNode) e.nextElement(); if(vjn.isUINode&&vjn.comp.bounds ().intersects(r)){ if(cntrDwn){ if(vjn.getSelected()) { vjn.setSelected(false); clearLite(getGraphics(),vjn.comp.bounds()); }else vjn.setSelected(true); } else vjn.setSelected(true); emptySelect = false; } } if(!emptySelect) { repaint(); container.theDesktop.vp_w.repaint(); } } void drawDrag(Graphics g,int i,int j,int k,int l){ for(Enumeration e = app.theContainer.nodes.elements(); e.hasMoreElements () ;) { VJNode vjn = (VJNode) e.nextElement(); if(vjn.getSelected()) g.drawRect (vjn.comp.bounds().x-1+xend- xbeg,vjn.comp.bounds().y-1+yend-ybeg, vjn.comp.bounds ().width+1,vjn.comp.bounds().height+1); } } void eraseDrag(Graphics g,int i,int j,int k,int l){ for(Enumeration e = app.theContainer.nodes.elements(); e.hasMoreElements() ;) { VJNode vjn = (VJNode) e.nextElement(); if(vjn.getselected ()) g.drawRect(vjn.comp.bounds().x-1+xend- xbeg,vjn.comp.bounds().y-1+yend-ybeg, vjn.comp.bounds().width+1,vjn.comp.bounds().height+1); } } void endDragAction (Graphics g,int i,int j,int k,int l){ for(Enumeration e = app.theContainer.nodes.elements.(); e.hasMoreElements() ;) { VJNode vjn = (VJNode) e.nextElement(); if(vjn.getSelected()){ clearLite(g, vjn.comp.bounds()); vjn.comp.move(vjn.comp.bounds().x+xend- xbeg,vjn.comp.bounds().y+yend-ybeg); } } } void containerPaint(Graphics g){ } void containerMouseMove(Event e){ } void containerKeyPress(Event e){ if(e.key==127) app.theContainer.doCut(); if(e.key==4) container.DUMP(); if(e.controlDown()){ switch(e.key){ case 3: System.out.println ("copy"); app.theContainer.doCopy(); break; case 24: System.out.println("cut"); app.theContainer.doCut(); break; case 22: System.out.println("paste"); app.theContainer.doPaste(); break; case 1: System.out.println("all"); app.theContainer.doSelectAll(); break; default: System.out.println("key"+e.key); break; } } } void containerMouseDown(Event e){ } void containerMouseUp(Event e){ if (grow) endGrow(e.x,e.y); } void containerMouseDrag(Event e){ if(grow) doGrow (e.x,e.y); } void containerDoubleClick(){ } boolean mouseDownSelects(Event e) { return true; } boolean mouseDownSelection(Event e){ VJNode vjn = onUIComponent (e.x,e.y); if(vjn==null){ if(!e.controlDown()) clearlight(getGraphics()); System.out.println("Noton a UI Component"); return false; } doUISelection (current_i,e.controlDown(),e.x,e.y); return true; } void mouseDownReset(Event e) { clearlight(getGraphics()); } boolean doMarquee() { return true; } public VJNode onUIComponent(int x, int y){ for(Enumeration e = app.theContainer.nodes.elements(); e.hasMoreElements();) { VJNode vjn = (VJNode) e.nextElement(); if(vjn.isUINode) { Rectangle r = vjn.comp.bounds(); r.x = r.x-4; r.y = r.y-4; r.width = r.width+8;

```
r.height = r.height+8; if(r.inside(x,y)){ current_i = vjn; return vjn; } } } return
null; } public boolean closeEnough(int fx, int fy,int x, int y,int epsilon){ return
x <fx+epsilon && x >fx-epsilon && y <fy+epsilon && y >fy- epsilon; } public int
getGrowType(VJNode vjn, int x, int y){ int top_y, mid_y,
bottom_y,left_x,mid_x,right_x; Rectangle r = vjn.comp.bounds(); top_y = r.y-4;
mid_y = r.y+(r.height+1)/2; bottom_y = r.y+r.height+4; left_x = r.x-4; mid_x = r.x+
(r.width+1)/2; right_x = r.x+r.width+4; if(closeEnough(left_x,top_y, x,y,4)) return
0; if(closeEnough(mid_x ,top_y, x,y,4)) return 1; if(closeEnough(right_x,top_y,
x,y,4)) return 2; if(closeEnough(left_x,mid_y, x,y,4)) return 3; if(closeEnough
(right_x,mid_y, x,y,4)) return 4; if(closeEnough(left_x,bottom_y,x,y,4)) return 5;
if(closeEnough(mid_x ,bottom_y,x,y,4)) return 6; if(closeEnough
(right_x,bottom_y,x,y,4))return 7; return 8; } public void doUISelection
(VJNodevjn,boolean cntDwn,int x, int y){ growType = getGrowType(vjn,x,y);
System.out.println("DoUI "+growType); if(!cntDwn){ if(!vjn.getSelected())
{ resetSelected(); vjn.setSelected(true); } if(growType==8) beginDrag(x,y); else
beginGrow(vjn,x,y); //current_component=i; repaint();
container.theDesktop.vp_w.repaint(); } else { if(vjn.getSelected()){ Rectangle r =
vjn.comp.bounds(); vjn.setSelected(false); clearLite(getGraphics(),r); for
(Enumeration e = app.theContainer.nodes.elements(); e.hasMoreElements();) { VJNode
vjn1 = (VJNode) e.nextElement(); if(vjn1.getSelected()){ repaint(); return; } } }
else { vjn.setSelected(true); container.theDesktop.vp_w.repaint(); repaint();
return; } } } public boolean bigEnough(){ //return xbeg-yend !=0 && ybeg-xbeg!=0;
return true; } public void beginGrow(VJNode vjn, int x, int y){ Rectangle r =
vjn.comp.bounds(); xbeg = x ; ybeg = y ; xend = x ; yend = y ; grow = true;
Graphics g = getGraphics(); g.setXORMode(Color.white); top = r.y; ytop=top; left =
r.x; xleft=left; bottom = r.y+vjn.comp.bounds().height;ybottom=bottom; right =
r.x+vjn.comp.bounds().width;xright = right; g.drawRect( left, top, r.width,
r.height ); } public void doGrow(int x, int y){ Graphics g = getGraphics();
g.setXORMode(Color.white); g.drawRect(left, top, right-left ,bottom-top); xend =
x ; yend = y ; switch(growType){ case 0: // top left top = ytop+(yend-ybeg); left =
xleft+(xend-xbeg);
```

Detailed Description Paragraph Table (10):

```
break; case 1: // top middle top = ytop+(yend-ybeg); break; case 2: // top right
top = ytop+(yend-ybeg); right = xright+(xend-xbeg); break; case 3: // middle left
left = xleft+(xend-xbeg); break; case 4: // middle right right = xright+(xend-
xbeg); break; case 5: // bottom left left = xleft+(xend-xbeg); bottom = ybottom+
(yend-ybeg); break; case 6: // bottom middle bottom = ybottom+(yend-ybeg); break;
case 7: // bottom right right = xright+(xend-xbeg); bottom = ybottom+(yend-ybeg);
break; } g.drawRect(left, top, right-left ,bottom-top); } public void endGrow(int
x, int y){ Graphics g = getGraphics(); g.setXORMode(Color.white); g.drawRect(left,
top, right-left ,bottom-top); xend = x ; yend = y ; clearLite(g,
current_i.comp.bounds()); current_i.comp.reshape(left,top, right-left ,bottom-
top); //if(current_i.compinstanceofVJChart) ((VJChart)(current_i.comp)).doResize
(left,top, right-left ,bottom-top); drawLite(g, current_i.comp.bounds()); grow =
false; } public void clearlight(Graphics g) { for(Enumeration e =
app.theContainer.nodes.elements(); e.hasMoreElements();) { VJNode vjn = (VJNode)
e.nextElement(); if(vjn.getSelected()&&vjn.isUINode){ clearLite(g,vjn.comp.bounds
()); vjn.setSelected(false); } } container.theDesktop.vp_w.repaint(); } public void
highlight(Graphics g) { for(Enumeration e = app.theContainer.nodes.elements();
e.hasMoreElements();) { VJNode vjn = (VJNode) e.nextElement(); if(vjn.getSelected()
&&vjn.isUINode){ drawLite(g,vjn.comp.bounds()); } } } public void drawLite(Graphics
g, Rectangle r) { int top_y, mid_y, bottom_y,left_x, mid_x,right_x; top_y = r.y;
mid_y = r.y+(r.height+1)/2-1; bottom_y=r.y+r.height; left_x =r.x; mid_x =r.x+
(r.width+1)/2-1; right_x =r.x+r.width; g.setColor(Color.red); g.fillRect(left_x-4,
top_y-4, 4, 4); g.fillRect(left_x-4, mid_y-2, 4, 4); g.fillRect(left_x-4, bottom_y,
4, 4); g.fillRect(right_x, top_y-4, 4, 4); g.fillRect(right_x, mid_y-2, 4, 4);
g.fillRect(right_x, bottom_y, 4, 4); g.fillRect(mid_x-2, top_y-4, 4, 4); g.fillRect
(mid_x-2, bottom_y, 4, 4); } public void clearLite(Graphics g, Rectangle r) { int
top_y, mid_y, bottom_y,left_x,mid_x,right_x; top_y = r.y; mid_y = r.y+
```

(r.height+1)/2-1; bottom_y=r.y+r.height; left_x =r.x; mid_x =r.x+(r.width+1)/2-1; right_x =r.x+r.width; g.clearRect(left_x-4, top_y-4, 4, 4); g.clearRect(left_x-4, mid_y-2, 4, 4); g.clearRect(left_x-4, bottom_y, 4, 4); g.clearRect(right_x, top_y-4, 4, 4); g.clearRect(right_x, mid_y-2, 4, 4); g.clearRect(right_x, bottom_y, 4, 4); g.clearRect(mid_x-2, top_y-4, 4, 4); g.clearRect(mid_x-2, bottom_y, 4, 4); } public void resetSelected() { } public synchronized VJNode newComponent(boolean cntrDwn,int x, int y){ int j,i,t,l,b,r; if(xbeg < x && ybeg < y) { t = ybeg; l = xbeg; b = y; r = x; } else if(xbeg < x && ybeg > y) { t = y; l = xbeg; b = ybeg; r = x; } else if(xbeg > x && ybeg < y) { t = ybeg; l = x; b = y; r = xbeg; } else { t = y; l = x; b = ybeg; r = xbeg; } if((r-1)<16 .vertline..vertline. (b-t)<16) { return null; } // resetSelected(); VJNode vjn; switch(current_ui_comp){ case 1: VJLabel vjl = new VJLabel(app); vjl.VJLabelInit(xOffset,yOffset); vjn = (VJNode) vjl; break; case 4: VJButton vjb = new VJButton(app); vjb.VJButtonInit (xOffset,yOffset); vjn = (VJNode) vjb; break; case 5: VJCheckbox vjcb = new VJCheckbox(app); vjcb.VJCheckboxInit(xOffset,yOffset); vjn = (VJNode) vjcb; break; case 6: VJChoice vjch = new VJChoice(app); vjch.VJChoiceInit(xOffset,yOffset); vjch.comp.reshape(l,t,r-l,b-t+l); vjn = (VJNode) vjch; break; case 7: VJList vjli = new VJList(app); vjli.VJListInit(xOffset,yOffset); vjli.comp.reshape(l,t,r-l,b-t+l); vjn = (VJNode) vjli; break; case 8: VJHScrollbar vjhsb = new VJHScrollbar (app); vjhsb.VJHScrollbarInit(xOffset,yOffset); vjn = (VJNode) vjhsb; break; case 9: VJVScrollbar vjvsb = new VJVScrollbar(app); vjvsb.VJVScrollbarInit (xOffset,yOffset); vjn = (VJNode) vjvsb; break; case 10: VJChart vjchart = new VJChart(app); vjchart.VJChartInit(xOffset,yOffset); vjn = (VJNode) vjchart; break; case 2: VJTextField vjt = new VJTextField(app); vjt.VJTextFieldInit (xOffset,yOffset); vjn = (VJNode) vjt; break; case 3: VJTextArea vjta = new VJTextArea(app); vjta.VJTextAreaInit(xOffset,yOffset); vjn = (VJNode) vjta; break; default: System.out.println("UNKNOWNTYPE!"); return null; } if(!cntrDwn) { container.theDesktop.vp_w.resetSelected(); } container.addNode((Object)vjn); if (yOffset>220) { yOffset = 100; xOffset = xOffset+60; } else yOffset = yOffset+40; vjn.comp.move(l,t); add(vjn.comp); validate(); vjn.comp.reshape(l,t,r-l,b-t); vjn.comp.show(); vjn.init(); vjn.propertiesEditor(); return vjn; } public void update(Graphics g){ paint(g); } public void paint(Graphics g) { highlight(g); VJDocument works in conjunction with "container.java." The source code for "container.java" appears below. importjava.util.*; importjava.awt.*; abstract class container extends Panel { boolean marquee; boolean drag; int xbeg,xend,ybeg,yend; int inset_h,inset_v; boolean firstTime = true; static long lastTime=0; VJ app; public container(VJ v) { super(); setLayout(null); app = v; marquee = false; drag = false; } public void beginMarquee(int x, int y){ //System.out.println ("BEGINMARQUEE"); xbeg = x ; ybeg = y ; xend = x ; yend = y ; marquee = true; Graphics g = getGraphics(); g.setXORMode(Color.white); g.drawRect(xbeg, ybeg, 0, 0); } public void doMarquee(int x, int y){ //System.out.println("DOMARQUEE"); Graphics g = getGraphics(); g.setXORMode(Color.white); if(xbeg < xend) { if(ybeg < yend) g.drawRect(xbeg, ybeg, xend - xbeg, yend - ybeg); else g.drawRect(xbeg, yend, xend - xbeg, ybeg - yend); } else { if(ybeg < yend) g.drawRect(xend, ybeg, xbeg - xend, yend - ybeg); else g.drawRect(xend, yend, xbeg - xend, ybeg - yend); } xend = x ; yend = y ; if(xbeg < xend) { if(ybeg < yend) g.drawRect(xbeg, ybeg, xend - xbeg, yend - ybeg); else g.drawRect(xbeg, yend, xend - xbeg, ybeg - yend); } else { if(ybeg < yend) g.drawRect(xend, ybeg, xbeg - xend, yend - ybeg);

Current US Cross Reference Classification (1):
717/107

CLAIMS:

15. A computer program embodied on a computer-readable medium for creating a collaborative relationship between objects, components or assemblies thereof that are part of an applet, which is resident on a hardware based server, utilizing a collaborative application server that is also resident on the hardware based server, said embodied program comprising: first software that registers each object, component or assembly for which collaboration is desired with the

application server as a registered object, component or assembly; second software that builds a record within the application server of all objects, components, assemblies that are to be collaborative; third software that receives information from each registered object, component or assembly that has been changed of the change and of the changed information; and fourth software that notifies each registered object, component or assembly that another object, component or assembly to which it is collaboratively linked has been changed.

16. The computer program embodied on a computer-readable medium as recited in claim 15 which additionally comprises fifth software for having the application server instantiate each registered object, component or assembly as derived from the applet and then link them in accordance with the specific collaboration desired so that a linked network of collaborated objects, components or assemblies can be maintained by the application server.

19. The computer program embodied on a computer-readable medium as recited in claim 15 which additionally comprises sixth software for notifying each registered object, component or assembly that another object, component or assembly to which it is collaboratively linked has been changed and for automatically updating each linked, but not yet changed collaborated object, component or assembly.

20. The computer program embodied on a computer-readable medium as recited in claim 17 which additionally comprises seventh software for notifying each registered object, component or assembly that another object, component or assembly to which it is collaboratively linked has been changed and for automatically updating each linked, but not yet changed collaborated object, component or assembly.

21. The computer program embodied on a computer-readable medium as recited in claim 18 which additionally comprises seventh software for notifying each registered object, component or assembly that another object, component or assembly to which it is collaboratively linked has been changed and for automatically updating each linked, but not yet changed collaborated object, component or assembly.

First Hit Fwd Refs
End of Result Set

☐ Generate Collection Print

L6: Entry 7 of 7 File: USPT Jun 15, 1999

DOCUMENT-IDENTIFIER: US 5913065 A
TITLE: System, method and article of manufacture for creating hierarchical folder components for use in a java application or applet

Detailed Description Text (8):
Another technology that has function and capability similar to JAVA is provided by Microsoft and its ActiveX technology, to give developers and Web designers the wherewithal to build dynamic content for the Internet and personal computers. ActiveX runs only the so-called Wintel platform (a combination of a version of Windows and an Intel microprocessor), as contrasted with Java which is a compile once, run anywhere language.

Detailed Description Text (9):
ActiveX includes tools for developing animation, 3-D virtual reality, video and other multimedia content. The tools use Internet standards, work on multiple platforms, and are being supported by over one hundred companies. The group's building blocks are called ActiveX Controls, small, fast components that enable developers to embed parts of software in hypertext markup language (HTML) pages. ActiveX Controls work with a variety of programming languages including Microsoft's Visual C++, Borland's Delphi, Microsoft's Visual Basic programming system and, in the future, Microsoft's development tool for Java, code named "Jakarta." ActiveX Technologies also includes ActiveX Server Framework, allowing developers to create server applications. One of ordinary skill in the art will readily recognize that ActiveX and ActiveX components could be substituted for JAVA and its components as their use is described herein without undue experimentation to practice the invention.

Detailed Description Text (17):
Once obtained from its web page or local storage, the VJ Tool is itself initialized as shown in block 308 and made ready for interaction with a user, see block 310. Initialization of the applet includes initialization of the desktop (VJContainer) and the web page view (VJDocument). VJContainer and VJDesktop are tightly coupled. VJContainer is a container API. The details of applet initialization for VJ Tool are included in VJContainer and VJDesktop.

Detailed Description Text (20):
The source code enabling the container editor for modifying properties of the container is presented below. When a new component is instantiated on either the physical or logical display, an optional customizer (edit) window is presented if a containerEditor method is defined for the component. The customizer (edit) window is defined in the template as a method corresponding to the customizer (edit) method. The method contains the logic facilitating the dynamic definition of properties of the component and the update of the properties based on user interaction with the customizer (edit) window. ##SPC2##

Detailed Description Text (27):
FIG. 5 also shows the physical view 500 of the VJ desktop. The logical view 402 is in the foreground on the right hand side of the drawing, while the physical view

500 is in the background on the left side of the screen. Physical view 500 is provided with palette containers 502 to 530 as follows: 502 is a select cursor button; 504 is a simple AWT label; 506 is a simple AWT text field; 508 is a simple AWT text area; 510 is a simple AWT button; 512 is a simple AWT checkbox; 514 is a simple AWT choice box; 516 is a simple AWT list; 518 is a simple AWT horizontal scroll bar; 520 is a simple AWT vertical scroll bar; 522 is a simple bar chart; 524 is a simple spreadsheet; 526 is a simple AWT panel; 528 is a calendar; and 530 is an animator. Logical view 402 is provided with palette containers 540 to 572 as follows: 540 is a select cursor button; 542 is a VJ folder or container; 544 is an adder; 546 is a bicopy component. The bicopy component acts as a multiplexor so that whatever is input on one pin of a bicopy element is output on its other pins. 548 is a test for equals; 550 is a constant (e.g. -1 or 3.1457 or "abc"); 552 is a random number generator; 554 is a counter; 556 is an URL opener; 558 is a splitter used to connect input/output pins to components that are either input or output elements; 560 is a two-dimensional grapher; 562 is a three-dimensional grapher; 564 is a delta three-dimensional grapher; 566 is an URL text viewer; 570 forwards input from any one of an element's input pins to an outpin pin thereof; 572 is a calculator; and 574 is a sound player. AWT is the Abstract Windowing Toolkit and contains the objects utilized for providing basic windowing functions in accordance with a preferred embodiment.

Detailed Description Text (32):
Liveness also indicates that the environment is tested as new components are instantiated. This means that immediate connections between components are permitted on the fly. There is no requirement that all possible connections are tried first to insure that nothing blows up. The outcome of this capability is that design creation and trial are integrated so that the results can be observed virtually at the same time that the interconnection of components is made. As noted later herein in connection with the description of FIG. 16, this is analogous to building a hardware prototype or test board by interconnecting, disconnecting and moving electronic components around the test board while the power is left on. As will be shown by the following example of Fahrenheit and Centigrade conversion, there is immediate feedback of component status and value information. This makes VJ Tool, in essence, a WYSIWIG development environment.

Detailed Description Text (68):
Decision block 1626 sends control back to polling if the pending connection has not yet been made. If the connection is completed, control for the ports involved is passed to decision block 1628 which decides if the attempted complete connection between the ports involved is valid. The type of ports involved and their functional affinity for each other will be checked and verified. Connections directly between an input port on one component will not be allowed to an input port on another component nor will a direct connection between an output port on a first container to an output port on a second container. Also forbidden are connections between ports on the same component regardless of their type.

Detailed Description Text (87):
FIG. 23 illustrates an edit screen for a folder component utilized to prepare a hierarchical component in accordance with a preferred embodiment. A Container 1 folder 2300 is instantiated in the logical desktop 2350 by dropping a folder component 2360 on the desktop. A customize or edit window 2310 pops up when the folder is instantiated. This edit window can be utilized to change the name of the folder and the number, location and type of active ports of the folder component. The Container 1 folder is logically represented by a window 2320 where the logic of the folder (hierachial component) can be created on the fly by adding or deleting components in the window 2320. The button 2330 is also shown in the physical desktop 2340.

Detailed Description Text (89):
FIG. 24 illustrates a hierarchical component creation customization in accordance

with a preferred embodiment. The button 2400 which was originally in the logical desktop 2420 has been moved into the Container 1 folder 2410. The internal connectors 2430C and 2440C are represented in the Container 1 folder 2410 and refer back to the logical desktop ports 2440L and 2430L of the folder 2450 in the logical desktop 2420. The ports can be used to transfer information into and out of a folder 2410. One of ordinary skill in the art will readily recognize that other components could be added to the Container 1 folder 2410 to create a more complex hierarchical component which could even include nested and recursive hierarchical components.

Detailed Description Paragraph Table (3):

```
import java.awt.*; public class VJNetPin extends VJNode { / / Attributes of this component static int instanceCount = 0; static Image normalImage; static Image selectedImage; final static String out = "out.sub.-- np.gif"; final static String in = "in.sub.-- np.gif"; final static String port0.sub.-- info = "input or output and object"; final static String port0.sub.-- name = "Pin 0"; final static String url.sub.-- name = "netpin.html"; final static String info.sub.-- name = "Connects components inside a container to a pin of the container"; VJ vj; netpinEditor edit=null; VJContainer theContainer; boolean connected = false; boolean requested = false; int theLocation; int requestTime = 0; int theConnection = -1; / / Constructor public VJNetPin(VJ v) { super(v); vj = v; VJNode dup() { return null; } public static void getImages (GIFFactoryf){ normalImage = f.GetGIF(out); selectedImage = f.GetGIF(in); } public void setContainer(VJContainerc) { theContainer = c; } public void setConnection(int c) { theConnection = c; } / / Component Initialization public void VJNetPinInit(int x.sub.-- pt, int y.sub.-- pt) { try { setNormalIcon(out); setSelectedIcon(in); setName("VJNetPin"); setComponent(null); setComponentURL(url.sub.-- name); setComponentInfo(info.sub.-- name); VJNodeInit(false,x.sub.-- pt,y.sub.-- pt,false); addPort(port0.sub.-- info,port0.sub.-- name,VJPort.InputOutput,VJPort.Sout hCente r); //Pin 0 setImages (normalImage,selectedImage); / / Pass references to the static images down to the node nodeRect = new Rectangle(x.sub.-- pt-3,y.sub.-- pt- 3,selectedImage.getWidth (vj.theContainer.theDesktop.vp.sub.-- w)+3,selecte dI mage.getHeight (vj.theContainer.theDesktop.vp.sub.-- w)+3); } catch(Exception e) { System.out.println(e); } } public int componentID() { return 6; } public void disconnecting(int port) { connected = false; } public void connecting(int port) { connected = true; if(requested) { vj.request(0,requestTime,this); requested = false; } } public void load(String s) { } public String save() { return ""; } public void reset() {} public void request(int port,int time) { / / what if theConnection <0? theContainer.requestOUT(theConnection,time); } public void requestIN(int time) { if(connected)vj.request(0,time,this); else { requestTime = time; requested = true; } } public void set(Object o,int port,int time) { if (theConnection>=0) theContainer.setOUT(o,theConnection,time); / /vj.request (0,request.sub.-- index0++,this); } public void setIN(Object o,int time) { vj.set (o,0,time,this); } public void propertiesEditor() { if(edit==null){ edit = new netpinEditor((Frame)(vj.theFrame),this); edit.pack(); edit.show(); } } public void init(){}; public void start(){}; public void stop(){}; public void destroy(){}; } class netpinEditor extends Frame { VJNetPin vjc; TextField tf; Button b; Button cancel; public netpinEditor(Frame parent,VJNetPin c) { super("Pin Editor"); setLayout(newBorderLayout()); add("North",new Label("Select a pin")); vjc = c; tf = new TextField(new Integer(vjc.theLocation).toString()); add("Center",tf); b = new Button("OK"); cancel = new Button("Cancel"); Panel sp = new Panel(); sp.add(b); sp.add(cancel); add("South",sp); } public boolean handleEvent(Eventevt) { / / System.out.println(evt.toString()); switch(evt.id){ case Event.ACTION.sub.-- EVENT: { if("OK".equals(evt.arg)) { vjc.theLocation = (Integer.valueOf(tf.getText ())).intValue(); / / vjc.theContainer.addNewPort(vjc, "fred","jim"); vjc.edit = null; dispose(); return true; } if("Cancel".equals(evt.arg)) { vjc.edit = null; dispose(); return true; } return false; } default: return false; } } }
```

Current US Original Classification (1):
717/107

Current US Cross Reference Classification (2):
717/109

Current US Cross Reference Classification (3):
717/118

CLAIMS:

9. The method according to claim 1 wherein the folder component object is further configured to perform an operation upon data received from a connected software component object through an editable port of the folder component object.

10. The method according to claim 9 wherein the folder component object is configured to forward data to a connected software component object through an editable port of the folder component object.

11. The method according to claim 1 wherein the folder component object is configured to forward data to a connected software component object through an editable port of the folder component object.

15. The method according to claim 14 including the step of creating a connection between a port on the first software component object and the folder component object.

16. The method according to claim 15 including the step of creating a connection between a port on the folder component object and a port on a second software component object that is outside of the folder component object such that information can be transmitted between the first software object and the second software object.

19. The method according to claim 1 which includes the step of connecting a port of the instantiated folder component to another software component object.

20. A system for creating hierarchical, object oriented folder component objects for use in a development environment for creating an object oriented applet or application, said system comprising:

a. a folder class configured to produce instantiated folder component objects having a folder editor for editing folder component object properties and editable ports effective to transfer data and software components between the folder component object and another software component object;

b. an object oriented software development environment configured to instantiate the folder component object from the folder class, the development environment being further configured to enable connections between the port and other software component objects instantiated in the development environment; and

c. an editor for editing the properties of the folder component to reflect user requirements.

28. The system of claim 27, wherein the first software component object is connected with a port of the instantiated folder component object.

29. The system of claim 28, wherein a port of the instantiated folder component object is connected is connected with a second software component object.

35. The system of claim 34, wherein the third software object is connected to a

port on the second instantiated folder component object such that the third software component object can communicate with the first software component object contained in the first instantiated object folder.

42. The computer program embodied on a computer-readable medium as recited in claim 36 including software for enabling connections between a port of the folder component object and another other software component object.

First Hit

L9: Entry 1 of 5 File: PGPB May 16, 2002

DOCUMENT-IDENTIFIER: US 20020059558 A1
TITLE: Coordination-centric framework for software design in a distributed environment

Current US Classification, US Primary Class/Subclass:
717/103

Summary of Invention Paragraph:
[0007] The coordination-centric framework is based on the philosophy that complex software systems should be built from reusable components connected through high-level coordination operators. The coordination-centric framework allows for dynamic object insertion and removal through the use of a general purpose runtime system for managing the interactions between objects.

Brief Description of Drawings Paragraph:
[0025] FIG. 12A is software system with two data transfer coordinators, each having constant message consumption and generation rules and each connected to a separate data-generating component and connected to the same data-receiving component.

Detail Description Paragraph:
[0051] FIG. 2 is component 100 further including a first coordination interface 200, a second coordination interface 202, and a third coordination interface 204. Coordination-centric design's components 100 provide the code-sharing capability of object-oriented inheritance through copying. Another aspect of object-oriented inheritance is polymorphism through shared interfaces. In object-oriented languages, an object's interface is defined by its methods. Although coordination-centric design's actions 104 are similar to methods in object-oriented languages, they do not define the interface for component 100. Components interact through explicit and separate coordination interfaces, in this figure coordination interfaces 200, 202, and 204. The shape of coordination interfaces 200, 202, and 204 determines the ways in which component 100 may be connected within a software system. The way coordination interfaces 200, 202, and 204 are connected to modes 102 and actions 104 within component 100 determines how the behavior of component 100 can be managed within a system. Systemwide behavior is managed through coordinators (see FIG. 4B and subsequent).

Detail Description Paragraph:
[0066] A port is a primitive connection point for interconnecting components. Each port is a five-tuple (T; A; Q; D; R) in which:

Detail Description Paragraph:
[0309] A POD, once inserted into a VINE, injects a set of coordinators. PODs are containers of strongly cohesive PEAs. PEAs are objects that are dedicated to performing a single, distinct task. PODs are similar to simple objects in CORBA. PEAS are typically highly modular and highly reusable software elements that require little to no knowledge about any other software elements in a system. A POD must allow dynamic insertion of PEAs. For example, a display is a POD, and its top-level windows are PEAs.

CLAIMS:

1. A system for implementing a modular software system in a distributed computing environment, the distributed computing environment having a plurality of networked processing resources, comprising: a first and a second general runtime component which are capable of communicating with each other; the first and second general runtime components each comprising a plurality of sockets for connecting to a lower level component and managing interactions between the lower level components; the first and second general runtime components each capable of operating on one of the plurality of networked processing resources; a first and a second container component for engaging one of the plurality of sockets of one of the general runtime components; the first container component implements a first predetermined functionality and comprises a first container coordinator for allowing the second container component to communicate with the first container component through their respective general runtime components; and a first and second functional component within one of the container components; the first functional component performs a second predetermined functionality and comprises a first functional coordinator for allowing the second functional component to communicate with the first functional component.

2. A system according to claim 1 wherein the first container component can be dynamically inserted into the first general runtime component, at which point the first container component introduces its first container coordinator into the first general runtime component.

3. A system according to claim 2 wherein the second container component may then establish a connection with the first container component via the first container coordinator.

4. A system according to claim 3 wherein the first functional component can be dynamically inserted into the first container component, at which point the first functional component can introduce its first functional coordinator into the first container component.

5. A system according to claim 4 wherein the second function component may then establish a connection with the first container component via the first container coordinator.

6. A system according to claim 5 wherein the first functional component may be dynamically terminated from the first container component.

8. A system according to claim 7 wherein the first container component may be dynamically terminated from the first general runtime component.

9. A system according to claim 8 wherein the first container component will enter a container termination phase upon being dynamically terminated and will terminate the first functional component prior to completing the container termination phase.

10. A method for implementing a modular software system on a distributed computing platform, the distributed computing platform having a plurality of connected processing resources, the method comprising: providing a first and a second general runtime component which are capable of operating on one of the plurality of processing resources and of communicating with each other, and each of which comprise a plurality of sockets for connecting to a lower level component and managing interactions between the lower level components; providing a first and a second container component for implementing a first and second predetermined functionality, respectively, dynamically connecting the first and second container components into the one of the plurality of sockets in the first and second general runtime components, respectively; with the first container component, inserting a

first container coordinator into the first general runtime component, thereby creating a connection point for the second container component to initiate communication with the first container component by connecting to the first container coordinator; providing a first and second functional component for performing a third and fourth predetermined functionality, respectively, within the first container component; dynamically introducing the first functional component into the first container component; and with the first functional component, inserting a first functional coordinator into the first container component, thereby creating a connection point for the second functional component to initiate communication with the first functional component by connecting to the first functional coordinator.

11. A method according to claim 10 further comprising: dynamically removing the first container component from the first general runtime component; in the first container component, terminating the first functional component.

First Hit Fwd Refs

Generate Collection Print

L9: Entry 3 of 5 File: USPT Dec 14, 1999

DOCUMENT-IDENTIFIER: US 6003037 A
TITLE: Smart objects for development of object oriented software

Abstract Text (1):
An improved object-oriented programming environment for facilitating creation of database management applications is disclosed. The programming environment provides a method and apparatus for establishing named connections between encapsulated, individually designed software components referred to as "smart objects," which communicate and act in a coordinated fashion as part of a finished software application. Connections between smart objects are referred to as "smart links." A basic set of smart links for coordinating a core group of smart objects is disclosed, and a substantial variety of database management applications can be created using the disclosed smart objects and smart containers. The core group of smart objects each include four common capabilities: the capability to initialize and destroy themselves, the capability to get and set attributes, the capability to add and remove smart links, and the capability to communicate using a standard communication interface. Individual smart objects have other capabilities in addition to the four common capabilities.

Brief Summary Text (6):
An improved object-oriented programming environment for creating database management applications is disclosed. The programming environment provides a method and apparatus for establishing named connections between encapsulated, individually designed software components referred to as "smart objects." The connections, which are referred to as "smart links," allow the smart objects to communicate and act in a coordinated fashion as part of a finished software application. A basic set of smart links for coordinating a core group of smart objects which provide tools to build a substantial variety of database management applications is disclosed.

Brief Summary Text (7):
The core group of smart objects each include four common capabilities: the capability to initialize and destroy themselves, the capability to get and set attributes, the capability to add and remove smart links, and the capability to communicate using a standard communication interface. Individual smart objects have other capabilities in addition to the four common capabilities. The core group of smart objects includes a container object, a navigation object, a tableIO object, a paging object, a query object, a viewer object, and a browser object. The query object provides a source for database records. The browser object provides presentation of tabular data in rows and columns. The viewer object views a single row of data. The navigation object provides a source for messages of a navigational nature, i.e., messages such as "Next record" and "Previous record." The tableIO object provides a source for record manipulation messages, i.e., messages such as "Save," "Update," "Undo changes," and "Delete." The paging object controls the visualization of unique pages of objects. The container object provides containment of other objects for grouping and encapsulation.

Detailed Description Text (2):
FIG. 1 illustrates an improved object-oriented programming environment for creating database management applications. The programming environment provides a method and

apparatus for establishing named connections, referred to as "smart links," between encapsulated, individually designed software components referred to as "smart objects." The smart links 10 allow the smart objects 12 to communicate and act in a coordinated fashion as part of a finished software application 14. A set of smart links sufficient for coordinating a core group of smart objects 12 for building a substantial variety of database management applications 14 is included in the programming environment.

Detailed Description Text (3):
The programming environment includes a user interface builder 16 which is employed to assemble smart objects in order to provide the database management application. The core group of smart objects includes a container object 18, a navigation object 20, a tableIO object 22, a paging object 24, a query object 26, a viewer object 28, a browser object 30 and a user defined object 32. Each smart object is an encapsulated procedure with particular capabilities. All smart objects have four common capabilities: the capability to initialize and destroy themselves, the capability to get and set attributes, the capability to add and remove smart links, and the capability to communicate using a standard communication interface. Individual smart objects have other capabilities in addition to the four common capabilities. The query object 26 provides a source for database records. The viewer object 28 provides presentation of a single row, or view, of data. The browser object 30 provides presentation of tabular data in rows and columns. The navigation object 20 provides a source for messages of a navigational nature, i.e., messages such as "Next record" and "Previous record." The tableIO object 22 provides a source for record manipulation messages, i.e., messages such as "Save," "Update," "Undo changes," and "Delete." The paging object 24 provides control of the unique pages, or collections, of objects. The container object 18 provides containment of other objects for grouping and encapsulation

Detailed Description Text (4):
A set of smart objects can be encapsulated as a group in the container object 18. The container grouping normally corresponds to a container visualization such as a Frame, Dialog Box, or Window, but can also be a strictly logical grouping. The container can then be used as a single object in another parent object. When the container is used in this manner, links can be established between the container and the parent. At runtime, a first link 34 from a SOURCE outside of the container and a second link 36 of the same type from the container to an object within it are merged automatically into a single link, thereby allowing the true SOURCE and TARGET for the set of behaviors defined by the link to communicate directly and transparently with each other without violating the integrity of the container as a logical object at the time the application components are assembled.

Detailed Description Text (5):
Other smart objects can be created from a combination of the predefined smart objects. For example, the container object 48 and the browser object 30 can be combined into a newly defined class of object which encompasses the capabilities and behavior of both of the parent objects. The newly defined objects can then be used with the predefined smart objects. User defined smart objects can be created using one of the predefined smart objects as a base or by using a newly defined and described smart object template.

Detailed Description Text (6):
Referring now to FIGS. 1, 2a-2g, the smart objects communicate and exchange data using the smart links 10. Various types of predefined smart links are provided within the programming environment, including a record link 38, a navigation link 40, a tableIO link 42, a state link 44, a group-assign link 46, a container link 48, and a page link 50. The record link 38 provides for the exchange of record pointers. The navigation link 40 is used for messages involving record positioning, i.e., record positioning messages such as "Next," "Previous," and "Last." The tableIO link 42 is used for messages involving record manipulation, i.e., messages

such as "Save," "Update," and "Delete." The state link 44 is used for transmission of object state information. The group-assign link 46 is used for synchronization of messages among multiple related objects. The container link 48 is used for messages involving object containment, i.e., messages such as "Hide" and "View." The page link 50 is used for management of multiple visual pages within the user interface. A pageN link can be used for management of messages to objects within a visual page with an interface. Other smart links can be created by modifying one of the predefined smart links.

Detailed Description Text (13):
A mapping of common event procedures to the link types across which such procedures are normally sent is built into the application execution mechanism. This mapping can be extended or modified for the purposes of any particular software application, and hence allows the developer to omit the link name when dispatching events in other objects, as: RUN notify `row-available.` This command dictates that the row-available event is to be invoked in any object connected to the present object by any of the links named in the event mapping.

Detailed Description Text (15):
Any smart object may be a target, source or both target and source for any link, provided respective procedures for targeting, sourcing or targeting and sourcing the link are present within the object. For example, an object may have a plurality of sources of different link types simultaneously, provided the sources support the appropriate targeting procedures. However, an object may have only one source of any given link type. Exemplary groupings are shown in FIGS. 2a-2g. A Record link 38 can connect any record source object, e.g., Query, Browser, with any record target objects, e.g., Viewer. A Navigation link 40 can connect any navigation source object, e.g., Navigation, Panel, with any navigation target object, e.g., Viewer, Browser. A TableIO link 42 can connect any tableIO source, e.g., TableIO Panel, with any tableIO target, e.g., Viewer. A State link 44 can connect any smart object, e.g., Viewer with any smart object, e.g., Query. A Group-assign link 46 can connect any tableIO target, e.g., Viewer, with any group assign target, e.g., Viewer. A Container link 48 can connect any container object, e.g., Window or Frame, with any object it has created, with or without visual representation, e.g., Viewer. Finally, a Page link 50 can connect any paging object, e.g., Folder, with any smart Container, e.g., Smart Window.

Detailed Description Text (16):
A grouping or other combination of smart objects can be treated as a single smart object entity. Any smart objects placed within a container object can transmit and receive messages through the container object transparently using a pass-through link. In such a case, smart objects can be grouped together visually, within a common container, while maintaining an absolute degree of independence relevant to object messaging.

Current US Cross Reference Classification (1):
717/100

Current US Cross Reference Classification (2):
717/116

CLAIMS:

1. A method for connecting objects with an interactive software development environment comprising the steps of:

creating at least one design-time instance of an object;

querying the at least one object for links supported by the at least one object;

providing a dialog to assist developers to assign links by allowing the user to select links from a list and to create new links;

assuming that, where a record link is present, the record link must be supported by both source and target objects; and

determining that either

a) matching database tables exist in both objects, or

b) that the record-source can provide some identification for the record-target,

before the link is acceptable.

2. An interactive software development environment for development of an object oriented application with selected structure comprising:

a plurality of objects of at least first and second types, said objects containing a body of source code common to each of said objects;

a plurality of links of at least first and second types, said links operative to interconnect said objects;

an advisor capable of querying said objects for lists of desired links to support and which can provide suggestions as to links to make; and

an advisor operative to provide a notice of detected logical errors in the structure of the application, error detection being based upon interconnection of said objects and said links, said advisor providing notice of any detected missing link prior to testing said application.

9. An object-oriented programming environment for creation of database management applications with a plurality of predefined objects comprising:

a navigation object which provides navigational messages;

a tableIO object which provides record-manipulation messages;

a paging object which provides visualization of unique pages of objects;

a query object which provides database records;

a viewer object which facilitates presentation of data fields and forms;

a browser object which facilitates presentation of tabular data in rows and columns; and

a container object capable of containing other said navigation, tableIO, paging, query, viewer and browser objects,

wherein said predefined objects each include the capability to initialize and destroy themselves.

16. The programming environment of claim 15 wherein said predefined links further include a container link which facilitates exchange of object containment messages.

L9: Entry 4 of 5 File: USPT Jun 22, 1999

DOCUMENT-IDENTIFIER: US 5915113 A
TITLE: Visual application partitioning for creating distributed object oriented applications

Brief Summary Text (22):
Partitioning is defined for the purposes of this invention as the technique of dividing a monolithic or standalone application into multiple interconnected components. Each component is called a partition. Dividing an application into partitions is called partitioning.

Brief Summary Text (25):
The invention also provides a method for creating a distributed application for an object oriented environment having a visual building component adapted to display an application design input by a user, the displayed application design having multiple program objects and connections between said objects representing method calls. The method consists of the computer implemented steps of initially defining an internal representation of the objects and connections of the displayed application design. In response to user definition of at least one partition boundary in the displayed application design, corresponding empty partition containers are defined in the internal representation. In response to user relocation of at least one program object across the at least one partition boundary in the displayed application design, the program object is redefined in the internal representation as a distributed object and its connections crossing said at least one partition boundary as distributed connections. In response to a user commit action, code is generated that includes middleware characteristics for every distributed connection defined in the internal representation.

Brief Summary Text (26):
The present invention is also directed to a software development tool for designing and coding a distributed application that is adaptable to interoperate with a visual building component for displaying an application as multiple program objects having connections between them representing method calls, for displaying discrete partitions having boundaries, and for permitting a user to relocate program objects across partition boundaries while maintaining the visual representation of connections between the objects. The tool includes a metadata generator for defining a current internal representation of the displayed application and for defining any connections crossing partition boundaries in the displayed application as distributed connections in the current internal representation, and a code generator for generating distributed interfaces for all distributed connections defined in the current internal representation. The code generator is activated by user action.

Brief Summary Text (27):
In addition, the invention is directed to a computer program product that comprises computer usable medium having new, useful and nonobvious combination of "computer readable program code means" embodied therein for creating a distributed application for an object oriented environment having a visual building component adapted to display an application design input by a user where the displayed application design has multiple program objects and connections between said

objects representing method calls. The computer readable program code means in the computer program product consists of computer readable program code means for causing the computer to initially define an internal representation of the objects and connections of the displayed application design, and computer readable program code means for causing the computer, in response to user definition of at least one partition boundary in the displayed application design, to define corresponding empty partition containers in the internal representation. The computer readable program product also includes computer readable program code means for causing the computer, in response to user relocation of at least one program object across said at least one partition boundary in the displayed application design, to redefine said at least one program object in the internal representation as a distributed object and its connections crossing said at least one partition boundary as distributed connections. The computer readable program product includes, in addition, computer readable program code means for causing the computer, in response to a user commit action, to generate code including middleware characteristics for every distributed connection defined in the internal representation.

Detailed Description Text (13):
The user can create new partitions or partition containers, which are initially empty, on the visual display (block 23). In the preferred embodiment a known technique is employed for creating the partition graphic of selecting a partition creation tool from a tool palette and selecting the creation of a partition at a designated place in the display. Visually, the partitions could be delineated by simple border outlines, different coloured areas, or other known techniques for differentiating areas on a visual display. The visual representation of the new partition container(s) is concurrently reflected in the internal model of the application (block 24). Because each new partition potentially represents a separate processor on which program objects for the application will reside, the user may set middleware defaults on creating the new partitions (blocks 25,26).

Detailed Description Text (14):
The user can begin moving objects to the new partitions on the visual display (block 27). In the preferred embodiment, the user moves the objects using a "drag-and-drop" technique that is implemented using known technology. Other possible techniques that could be implemented to instantiate an object in a partition include clicking on an object and then on an empty partition container; selecting transfer locations for objects through listbox entries, etc. When an object is moved to a new partition, its connections with other objects (the connecting lines on FIG. 2) are maintained on the visual display.

Current US Original Classification (1):
717/109

Current US Cross Reference Classification (5):
717/103

Current US Cross Reference Classification (6):
717/105

CLAIMS:

3. A method for creating a distributed application for an object oriented environment having a visual building component adapted to display an application design input by a user, the displayed application design having multiple program objects and connections between said objects representing method calls, the method comprising the computer-implemented steps of:

initially defining an internal representation of the objects and connections of the displayed application design;

in response to user definition of at least one partition boundary in the displayed application design, defining corresponding empty partition containers in the internal representation;

in response to user relocation of at least one program object across said at least one partition boundary in the displayed application design, redefining said at least one program object in the internal representation as a distributed object and its connections crossing said at least one partition boundary as distributed connections; and

in response to a user commit action, generating code including middleware characteristics for every distributed connection defined in the internal representation.

6. A method, according to claim 3, further comprising the step of defining middleware defaults setting for said empty partition containers in response to user selection of middleware protocol options in the displayed application.

9. A software development tool for designing and coding a distributed application, the tool being adaptable to interoperate with a visual building component for displaying an application as multiple program objects having connections between them representing method calls, for displaying discrete partitions having boundaries, and for permitting a user to relocate program objects across partition boundaries while maintaining the visual representation of connections between the objects, the tool comprising:

a metadata generator for defining a current internal representation of the displayed application and for defining any connections crossing partition boundaries in said displayed application as distributed connections in the current internal representation; and

a code generator for generating distributed interfaces for all distributed connections defined in the current internal representation, said code generator being activated by user action.

12. A computer program product comprising:

a computer usable medium having computer readable program code means embodied therein for creating a distributed application for an object oriented environment having a visual building component adapted to display an application design input by a user, the displayed application design having multiple program objects and connections between said objects representing method calls, the computer readable program code means in said computer program product comprising:

computer readable program code means for causing the computer to initially define an internal representation of the objects and connections of the displayed application design;

computer readable program code means for causing the computer, in response to user definition of at least one partition boundary in the displayed application design, to define corresponding empty partition containers in the internal representation;

computer readable program code means for causing the computer, in response to user relocation of at least one program object across said at least one partition boundary in the displayed application design, to redefine said at least one program object in the internal representation as a distributed object and its connections crossing said at least one partition boundary as distributed connections; and

computer readable program code means for causing the computer, in response to a

user commit action, to generate code including middleware characteristics for every distributed connection defined in the internal representation.